

# A composite spectral scheme for variable coefficient Helmholtz problems

*P.G. Martinsson, Department of Applied Mathematics, University of Colorado at Boulder  
May 31, 2012*

**Abstract:** A discretization scheme for variable coefficient Helmholtz problems on two-dimensional domains is presented. The scheme is based on high-order spectral approximations and is designed for problems with smooth solutions. The resulting system of linear equations is solved using a direct solver with  $O(N^{1.5})$  complexity for the pre-computation and  $O(N \log N)$  complexity for the solve. The fact that the solver is direct is a principal feature of the scheme, since iterative methods tend to struggle with the Helmholtz equation. Numerical examples demonstrate that the scheme is fast and highly accurate. For instance, using a discretization with 12 points per wave-length, a Helmholtz problem on a domain of size  $100 \times 100$  wavelengths was solved to ten correct digits. The computation was executed on an office desktop; it involved 1.6M degrees of freedom and required 100 seconds for the pre-computation, and 0.3 seconds for the actual solve.

## 1. INTRODUCTION

The paper describes a technique for constructing an approximate solution to the variable coefficient Helmholtz equation

$$(1.1) \quad \begin{cases} -\Delta u(\mathbf{x}) - \kappa^2 (1 - b(\mathbf{x})) u(\mathbf{x}) = 0 & \mathbf{x} \in \Omega, \\ u(\mathbf{x}) = f(\mathbf{x}) & \mathbf{x} \in \Gamma, \end{cases}$$

where  $\Omega$  is a rectangular domain with boundary  $\Gamma$ , where the Helmholtz parameter  $\kappa$  is real, and where  $b$  is a given smooth scattering potential. The scheme can straight-forwardly be adapted to handle other variable coefficient elliptic problems, as well as free space scattering problems in  $\mathbb{R}^2$ . The primary limitation of the method is that it requires the solution  $u$  to be smooth in  $\Omega$ .

The equation (1.1) is discretized via a composite spectral scheme. The domain  $\Omega$  is split into small square (or rectangular) patches. On each patch, the solution  $u$  is represented via tabulation on a tensor product grid of Chebyshev points, see Figure 1. The Laplace operator is approximated via a spectral differentiation matrix acting on each local grid, and then equation (1.1) is enforced strongly at all tabulation nodes in the interior of each patch. To glue patches together, continuity of both the potential  $u$  and its normal derivative are enforced at the spectral interpolation nodes on the boundaries between patches.

The discretization scheme is combined with a direct solver for the resulting linear system. The fact that the solver is direct rather than iterative is a principal feature of the scheme, since iterative solvers tend to struggle for Helmholtz problems of the kind considered here [2]. The direct solver organizes the patches in the discretization into a binary tree of successively larger patches. The solver then involves two stages, one that involves an upwards pass, and one that involves a downwards pass:

- (1) A *pre-computation stage* where an approximation to the solution operator for (1.1) is computed. This is done via a single sweep of the hierarchical tree, going from smaller patches to larger. For each leaf in the tree, a local solution operator, and an approximation to the Dirichlet-to-Neumann (DtN) map for the patch are constructed. For a parent node in the tree, a local solution operator and a local DtN operator are computed from an equilibrium equation formed using the DtN operators of the children of the patch. The pre-computation stage has asymptotic complexity  $O(N^{1.5})$ .
- (2) A *solve stage* that takes as input a vector of Dirichlet data tabulated on  $\Gamma$ , and constructs tabulated values of  $u$  at all internal grid points. The solve stage involves a single downwards

sweep through the hierarchical tree of patches, going from larger patches to smaller. The solve stage has asymptotic complexity  $O(N \log N)$ .

Numerical experiments indicate that the spectral convergence of the method makes it both highly accurate and computationally efficient. For instance, the equation (1.1) was solved for a box whose size exceeded  $100 \times 100$  wave-lengths in less than 2 minutes on a standard office laptop. A 20-th order spectral scheme with 12 points per wave-length was used in the local approximation on the patches. The resulting solution was accurate to between 7 and 10 digits, depending on the nature of the scattering potential  $b$  in (1.1). The discretization used a total of  $N = 1.6 \cdot 10^6$  degrees of freedom. The computational time was dominated by the pre-computation stage; the actual solve stage took only 0.3 seconds. This makes the scheme particularly powerful in situations where an equation such as (1.1) needs to be solved for a sequence of different boundary functions  $f$ .

The scheme proposed is conceptually related to a direct solver for the Lippman-Schwinger equation proposed in 2002 by Yu Chen [1]. The schemes are different in that the method proposed here is not based on a Lippman-Schwinger formulation, and uses spectral approximations on the smallest patches in the hierarchical tree. Comparing the efficiencies of the two schemes is difficult since the paper [1] does not report numerical results, and we have been unable to find reports of implementations of the scheme. The scheme proposed here is also conceptually related to the classical nested dissection algorithm for finite element and finite difference matrices [3], and to recently proposed  $O(N^{1.5})$  direct solvers for BIEs on surfaces in 3D [4].

For clarity, the current paper focusses on the simple boundary value problem (1.1) involving the Helmholtz elliptic operator and Dirichlet boundary data. The scheme can with trivial modifications be applied to more general elliptic operators

$$-c_{11}(\mathbf{x})[\partial_1^2 u](\mathbf{x}) - 2c_{12}(\mathbf{x})[\partial_1 \partial_2 u](\mathbf{x}) - c_{22}(\mathbf{x})[\partial_2^2 u](\mathbf{x}) + c_1(\mathbf{x})[\partial_1 u](\mathbf{x}) + c_2(\mathbf{x})[\partial_2 u](\mathbf{x}) + c(\mathbf{x})u(\mathbf{x}) = 0,$$

coupled with Dirichlet, Neumann, or mixed boundary data. It has for instance been successfully tested on convection-diffusion problems that are strongly dominated (by a factor of  $10^4$ ) by the convection term, see Section 6.4. Moreover, the scheme can with minor modifications be applied to a free space scattering problem such as

$$(1.2) \quad -\Delta u(\mathbf{x}) - \kappa^2(1 - b(\mathbf{x}))u(\mathbf{x}) = f(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^2,$$

coupled with appropriate radiation conditions at infinity. A standard assumption is that  $f$  is supported outside of some (bounded) square region  $\Omega$  while the smooth function  $b$  is supported inside  $\Omega$ . The scheme described in this note computes the DtN operator for (1.2) on  $\Omega$ . The DtN operator for the exterior domain  $\Omega^c$  can be computed via Boundary Integral Equation techniques; and by combining the two, one can solve the free space scattering problem, see Section 7.3.

The method proposed has a vulnerability in that it crucially relies on the existence of DtN operators for all patches in the hierarchical tree. This can be problematic due to resonances: For certain wave-numbers  $\kappa$ , there exist non-trivial solutions that have zero Dirichlet boundary data. We have found that in practice, this problem almost never arises when processing domains that are a couple of hundred wave-lengths or less in size. Moreover, if a resonant patch should be encountered, this will be detected and counter-measures can be taken, see Sections 5.3 and 7.5.

The asymptotic complexity of the proposed method is  $O(N^{1.5})$ . For the case where the wave-number  $\kappa$  is increased as  $N$  grows to keep a constant number of discretization points per wave-length (i.e.  $\kappa \sim N^{0.5}$ ), we do not know how to improve the complexity. However, for the case where the wave-number is kept constant as  $N$  increases,  $O(N)$  complexity can very likely be attained by exploiting internal structure in the DtN operators. The resulting scheme would be a spectral version of recently published accelerated nested dissection schemes such as [8, 10, 12].

An early version of the work reported was published on arXiv as [9].

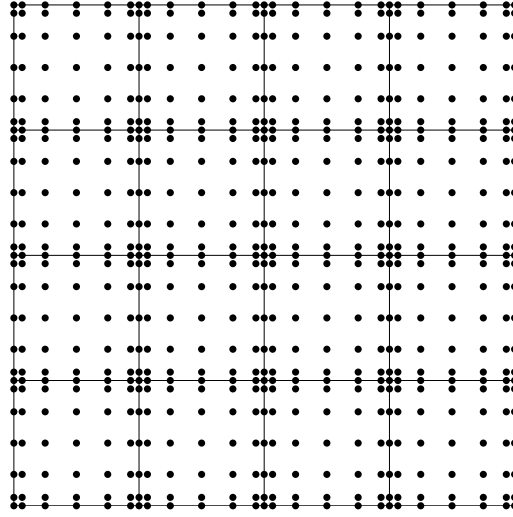


FIGURE 1. The box  $\Omega = [0, 1]^2$  is split into  $4 \times 4$  leaf boxes, and a Cartesian grid of Chebyshev nodes is placed on each leaf box. The figure shows local grids of size  $7 \times 7$  for clarity; in actual computations, local grids of size  $21 \times 21$  were typically used.

The paper is organized as follows: Section 2 introduces notation and lists some classical material on spectral interpolation and differentiation. Section 3 describes how to compute the solution operator and the DtN operator for a leaf in tree (which is discretized via a single tensor-product grid of Chebyshev nodes). Section 4 describes how the DtN operator for a larger patch consisting of two small patches can be computed if the DtN operators for the smaller patches are given. Section 5 describes the full hierarchical scheme. Section 6 reports the results of some numerical experiments. Section 7 describes how the scheme can be extended to more general situations.

## 2. PRELIMINARIES — SPECTRAL DIFFERENTIATION

This section introduces notation for spectral differentiation on tensor product grids of Chebyshev nodes on the square domain  $[-a, a]^2$ . This material is classical, see, e.g., Trefethen [11]. (While we restrict attention to square boxes here, all techniques generalize trivially to rectangular boxes of moderate aspect ratios.)

Let  $p$  denote a positive integer. The *Chebyshev nodes* on  $[-a, a]$  are the points

$$t_i = a \cos((i-1)\pi/(p-1)), \quad i = 1, 2, 3, \dots, p.$$

Let  $\{\mathbf{x}_k\}_{k=1}^{p^2}$  denote the set of points of the form  $(t_i, t_j)$  for  $1 \leq i, j \leq p$ . Let  $\mathcal{P}_p$  denote the linear space of sums of tensor products of polynomials of degree  $p-1$  or less.  $\mathcal{P}_p$  has dimension  $p^2$ . Given a vector  $\mathbf{u} \in \mathbb{R}^{p^2}$ , there is a unique function  $u \in \mathcal{P}_p$  such that  $u(\mathbf{x}_k) = \mathbf{u}(k)$  for  $1 \leq k \leq p^2$ . (A reason Chebyshev nodes are of interest is that for any fixed  $\mathbf{x} \in [-a, a]^2$ , the map  $\mathbf{u} \mapsto u(\mathbf{x})$  is stable.) Now define  $\mathbf{D}$ ,  $\mathbf{E}$ , and  $\mathbf{L}$  as the unique  $p^2 \times p^2$  matrices such that

$$(2.1) \quad [\mathbf{D}\mathbf{u}](k) = [\partial_1 u](\mathbf{x}_k), \quad k = 1, 2, 3, \dots, p^2,$$

$$(2.2) \quad [\mathbf{E}\mathbf{u}](k) = [\partial_2 u](\mathbf{x}_k), \quad k = 1, 2, 3, \dots, p^2,$$

$$(2.3) \quad [\mathbf{L}\mathbf{u}](k) = [-\Delta u](\mathbf{x}_k), \quad k = 1, 2, 3, \dots, p^2.$$

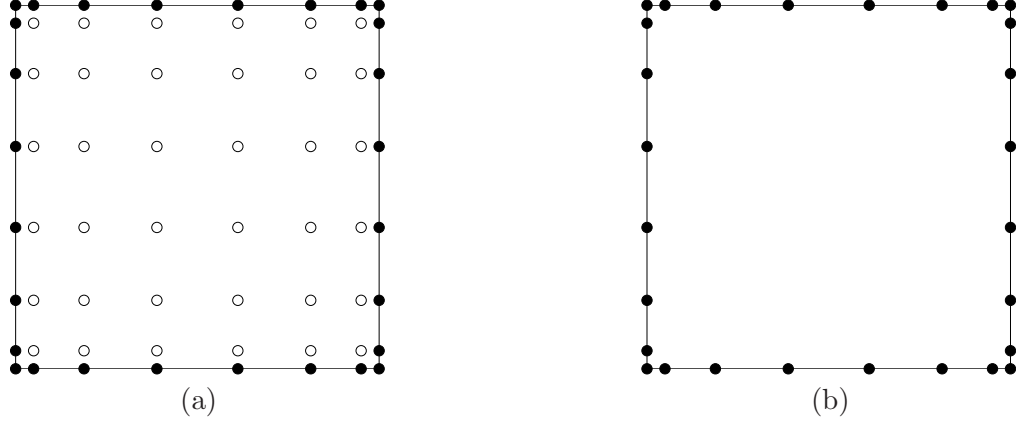


FIGURE 2. Notation for the leaf computation in Section 3. (a) A leaf before elimination of interior (white) nodes. (b) A leaf after elimination of interior nodes.

### 3. LEAF COMPUTATION

This section describes the construction of a discrete approximation to the Dirichlet-to-Neumann operator associated with the boundary value problem (1.1) for a square patch  $\Omega$ . We discretize (1.1) via a spectral method on a tensor product grid of Chebyshev nodes on  $\Omega$ . In addition to the DtN operator, we also construct a solution operator to (1.1) that maps the Dirichlet data on the nodes on the boundary of  $\Omega$  to the value of  $u$  at all internal interpolation nodes.

**3.1. Notation.** Let  $\Omega$  denote a square patch. Let  $\{\mathbf{x}_k\}_{k=1}^{p^2}$  denote the nodes in a tensor product grid of  $p \times p$  Chebyshev nodes. Partition the index set

$$\{1, 2, \dots, p^2\} = I_e \cup I_i$$

in such a way that  $I_e$  contains all nodes on the boundary of  $\Omega$ , and  $I_i$  denotes the set of interior nodes, see Figure 2(a). Let  $u$  be a function that satisfies (1.1) on  $\Omega$  and let

$$\mathbf{u} = [u(\mathbf{x}_k)]_{k=1}^{p^2}, \quad \mathbf{v} = [\partial_1 u(\mathbf{x}_k)]_{k=1}^{p^2}, \quad \mathbf{w} = [\partial_2 u(\mathbf{x}_k)]_{k=1}^{p^2},$$

denote the vectors of samples of  $u$  and its partial derivatives. We define the short-hands

$$\mathbf{u}_i = \mathbf{u}(I_i), \quad \mathbf{v}_i = \mathbf{v}(I_i), \quad \mathbf{w}_i = \mathbf{w}(I_i), \quad \mathbf{u}_e = \mathbf{u}(I_e), \quad \mathbf{v}_e = \mathbf{v}(I_e), \quad \mathbf{w}_e = \mathbf{w}(I_e).$$

Let  $\mathbf{L}$ ,  $\mathbf{D}$ , and  $\mathbf{E}$  denote spectral differentiation matrices corresponding to the operators  $-\Delta$ ,  $\partial_1$ , and  $\partial_2$ , respectively (see Section 2). We use the short-hand

$$\mathbf{D}_{i,e} = \mathbf{D}(I_i, I_e)$$

to denote the part of the differentiation matrix  $\mathbf{D}$  that maps exterior nodes to interior nodes, etc.

**3.2. Equilibrium condition.** The operator (1.1) is approximated via the matrix

$$\mathbf{A} = -\mathbf{L} - \kappa^2 \text{diag}(\mathbf{b}),$$

where  $\mathbf{b}$  denotes the vector of pointwise values of  $b$ ,

$$\mathbf{b} = [b(\mathbf{x}_k)]_{k=1}^{p^2}.$$

The equation we enforce on  $\Omega$  is that the vector  $\mathbf{A} \mathbf{u}$  should evaluate to zero *at all internal nodes*,

$$(3.1) \quad \mathbf{A}_{i,i} \mathbf{u}_i + \mathbf{A}_{i,e} \mathbf{u}_e = \mathbf{0},$$

where

$$\mathbf{A}_{i,i} = \mathbf{A}(I_i, I_i), \quad \mathbf{A}_{i,e} = \mathbf{A}(I_i, I_e).$$

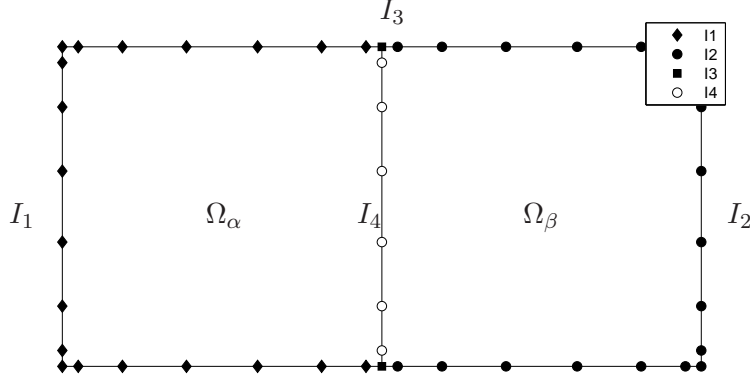


FIGURE 3. Notation for the merge operation described in Section 4. The rectangular domain  $\Omega$  is formed by two squares  $\Omega_\alpha$  and  $\Omega_\beta$ . The sets  $I_1$ ,  $I_2$ , and  $I_3$  form the exterior nodes (black), while  $I_4$  consists of the interior nodes (white).

Solving (3.1) for  $\mathbf{u}_i$ , we obtain

$$(3.2) \quad \mathbf{u}_i = \mathbf{U} \mathbf{u}_e,$$

where

$$(3.3) \quad \mathbf{U} = -(\mathbf{A}_{i,i})^{-1} \mathbf{A}_{i,e}.$$

**3.3. Constructing the DtN operator.** Let  $\mathbf{V}$  and  $\mathbf{W}$  denote the matrices that map boundary values of the potential to boundary values of  $\partial_1 u$  and  $\partial_2 u$ . These are constructed as follows: Given the potential  $\mathbf{u}_e$  on the boundary, we reconstruct the potential  $\mathbf{u}_i$  in the interior via (3.2). Then, since the potential is known on all Chebyshev nodes in  $\Omega$ , we can determine the gradient on the boundary  $\{\mathbf{v}_e, \mathbf{w}_e\}$  via spectral differentiation on the entire domain. To formalize, we find

$$\mathbf{v}_e = \mathbf{D}_{e,e} \mathbf{u}_e + \mathbf{D}_{e,i} \mathbf{u}_i = \mathbf{D}_{e,e} \mathbf{u}_e + \mathbf{D}_{e,i} \mathbf{U} \mathbf{u}_e = \mathbf{V} \mathbf{u}_e,$$

where

$$(3.4) \quad \mathbf{V} = \mathbf{D}_{e,e} + \mathbf{D}_{e,i} \mathbf{U}.$$

An analogous computation for  $\mathbf{w}_e$  yields

$$(3.5) \quad \mathbf{W} = \mathbf{E}_{e,e} + \mathbf{E}_{e,i} \mathbf{U}.$$

#### 4. MERGE OPERATION

Let  $\Omega$  denote a rectangular domain consisting of the union of the two smaller rectangular domains,

$$\Omega = \Omega_\alpha \cup \Omega_\beta,$$

as shown in Figure 3. Moreover, suppose that approximations to the DtN operators for  $\Omega_\alpha$  and  $\Omega_\beta$  are available. (Represented as matrices that map boundary values of  $u$  to boundary values of  $\partial_1 u$  and  $\partial_2 u$ .) This section describes how to compute a solution operator  $\mathbf{U}$  that maps the value of a function  $u$  that is tabulated on the boundary of  $\Omega$  to the values of  $u$  on interpolation nodes on the internal boundary, as well as operators  $\mathbf{V}$  and  $\mathbf{W}$  that map boundary values of  $u$  on the boundary of  $\Omega$  to values of the  $\partial_1 u$  and  $\partial_2 u$  tabulated on the boundary.

**4.1. Notation.** Let  $\Omega$  denote a box with children  $\Omega_\alpha$  and  $\Omega_\beta$ . For concreteness, let us assume that  $\Omega_\alpha$  and  $\Omega_\beta$  share a vertical edge. We partition the points on  $\partial\Omega_\alpha$  and  $\partial\Omega_\beta$  into four sets:

- $I_1$  Boundary nodes of  $\Omega_\alpha$  that are not boundary nodes of  $\Omega_\beta$ .
- $I_2$  Boundary nodes of  $\Omega_\beta$  that are not boundary nodes of  $\Omega_\alpha$ .
- $I_3$  The two nodes that are boundary nodes of  $\Omega_\alpha$ , of  $\Omega_\beta$ , and also of the union box  $\Omega$ .
- $I_4$  Boundary nodes of both  $\Omega_\alpha$  and  $\Omega_\beta$  that are *not* boundary nodes of the union box  $\Omega$ .

Figure 3 illustrates the definitions of the  $I_j$ 's. Let  $u$  denote a function such that

$$-\Delta u(\mathbf{x}) - \kappa^2 b(\mathbf{x}) u(\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega,$$

and let  $\mathbf{u}_j$ ,  $\mathbf{v}_j$ ,  $\mathbf{w}_j$  denote the values of  $u$ ,  $\partial_1 u$ , and  $\partial_2 u$ , restricted to the nodes in the set “ $j$ ”. Moreover, set

$$(4.1) \quad \mathbf{u}_i = \mathbf{u}_4, \quad \text{and} \quad \mathbf{u}_e = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \end{bmatrix}.$$

Finally, let  $\mathbf{V}^\alpha$ ,  $\mathbf{W}^\alpha$ ,  $\mathbf{V}^\beta$ ,  $\mathbf{W}^\beta$  denote the operators that map potential values on the boundary to values of  $\partial_1 u$  and  $\partial_2 u$  on the boundary for the boxes  $\Omega_\alpha$  and  $\Omega_\beta$ . We partition these matrices according to the numbering of nodes in Figure 3,

$$(4.2) \quad \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_3 \\ \mathbf{v}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{V}_{1,1}^\alpha & \mathbf{V}_{1,3}^\alpha & \mathbf{V}_{1,4}^\alpha \\ \mathbf{V}_{3,1}^\alpha & \mathbf{V}_{3,3}^\alpha & \mathbf{V}_{3,4}^\alpha \\ \mathbf{V}_{4,1}^\alpha & \mathbf{V}_{4,3}^\alpha & \mathbf{V}_{4,4}^\alpha \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_3 \\ \mathbf{u}_4 \end{bmatrix}, \quad \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_3 \\ \mathbf{w}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{W}_{1,1}^\alpha & \mathbf{W}_{1,3}^\alpha & \mathbf{W}_{1,4}^\alpha \\ \mathbf{W}_{3,1}^\alpha & \mathbf{W}_{3,3}^\alpha & \mathbf{W}_{3,4}^\alpha \\ \mathbf{W}_{4,1}^\alpha & \mathbf{W}_{4,3}^\alpha & \mathbf{W}_{4,4}^\alpha \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_3 \\ \mathbf{u}_4 \end{bmatrix},$$

and

$$(4.3) \quad \begin{bmatrix} \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{v}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{V}_{2,2}^\beta & \mathbf{V}_{2,3}^\beta & \mathbf{V}_{2,4}^\beta \\ \mathbf{V}_{3,2}^\beta & \mathbf{V}_{3,3}^\beta & \mathbf{V}_{3,4}^\beta \\ \mathbf{V}_{4,2}^\beta & \mathbf{V}_{4,3}^\beta & \mathbf{V}_{4,4}^\beta \end{bmatrix} \begin{bmatrix} \mathbf{u}_2 \\ \mathbf{u}_3 \\ \mathbf{u}_4 \end{bmatrix}, \quad \begin{bmatrix} \mathbf{w}_2 \\ \mathbf{w}_3 \\ \mathbf{w}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{W}_{2,2}^\beta & \mathbf{W}_{2,3}^\beta & \mathbf{W}_{2,4}^\beta \\ \mathbf{W}_{3,2}^\beta & \mathbf{W}_{3,3}^\beta & \mathbf{W}_{3,4}^\beta \\ \mathbf{W}_{4,2}^\beta & \mathbf{W}_{4,3}^\beta & \mathbf{W}_{4,4}^\beta \end{bmatrix} \begin{bmatrix} \mathbf{u}_2 \\ \mathbf{u}_3 \\ \mathbf{u}_4 \end{bmatrix}.$$

**4.2. Equilibrium condition.** Suppose that we are given a tabulation of boundary values of a function  $u$  that satisfies (1.1) on  $\Omega$ . In other words, we are given the vectors  $\mathbf{u}_1$ ,  $\mathbf{u}_2$ , and  $\mathbf{u}_3$ . We can then reconstruct the values of the potential on the interior boundary (tabulated in the vector  $\mathbf{u}_4$ ) by using information in (4.2) and (4.3). Simply observe that there are two equations specifying the normal derivative across the internal boundary (tabulated in  $\mathbf{v}_4$ ), and combine these equations:

$$\mathbf{V}_{4,1}^\alpha \mathbf{u}_1 + \mathbf{V}_{4,3}^\alpha \mathbf{u}_3 + \mathbf{V}_{4,4}^\alpha \mathbf{u}_4 = \mathbf{V}_{4,2}^\beta \mathbf{u}_2 + \mathbf{V}_{4,3}^\beta \mathbf{u}_3 + \mathbf{V}_{4,4}^\beta \mathbf{u}_4.$$

Solving for  $\mathbf{u}_4$  we get

$$(4.4) \quad \mathbf{u}_4 = (\mathbf{V}_{4,4}^\alpha - \mathbf{V}_{4,4}^\beta)^{-1} (\mathbf{V}_{4,2}^\beta \mathbf{u}_2 + \mathbf{V}_{4,3}^\beta \mathbf{u}_3 - \mathbf{V}_{4,1}^\alpha \mathbf{u}_1 - \mathbf{V}_{4,3}^\alpha \mathbf{u}_3).$$

Now set

$$(4.5) \quad \mathbf{U} = (\mathbf{V}_{4,4}^\alpha - \mathbf{V}_{4,4}^\beta)^{-1} [-\mathbf{V}_{4,1}^\alpha \mid \mathbf{V}_{4,2}^\beta \mid \mathbf{V}_{4,3}^\beta - \mathbf{V}_{4,3}^\alpha],$$

to find that (4.4) is (in view of (4.1)) precisely the desired formula

$$(4.6) \quad \mathbf{u}_i = \mathbf{U} \mathbf{u}_e.$$

The net effect of the merge operation is to eliminate the interior tabulation nodes in  $\Omega_\alpha$  and  $\Omega_\beta$  so that only boundary nodes in the union box  $\Omega$  are kept, as illustrated in Figure 4.

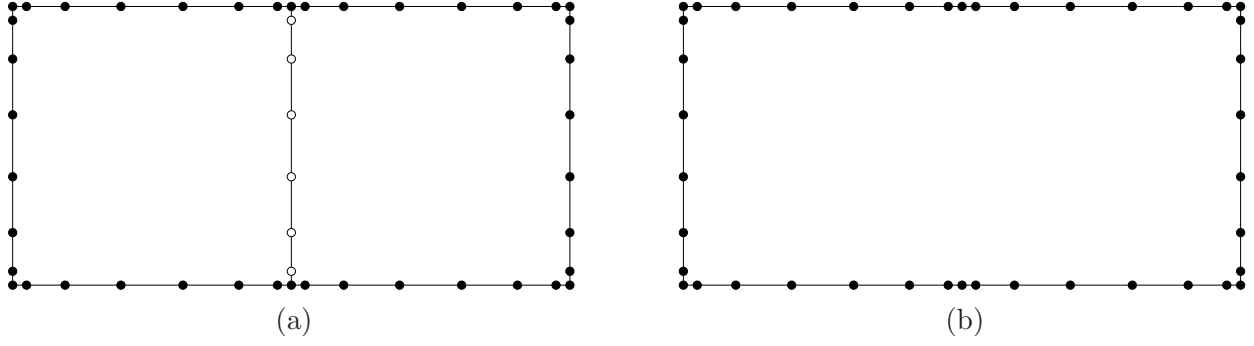


FIGURE 4. Merge operation for two small boxes to form a new large box. (a) Before elimination of interior (white) nodes. (b) After elimination of interior nodes.

**4.3. Constructing the DtN operators for the union box.** We will next build a matrix  $\mathbf{V}$  that constructs the derivative  $\partial_1 u$  on  $\partial\Omega$  given values of  $u$  on  $\partial\Omega$ . In other words

$$\begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \end{bmatrix} = \mathbf{V} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \end{bmatrix}.$$

To this end, observe from (4.2) and (4.3) that

$$(4.7) \quad \mathbf{v}_1 = \mathbf{V}_{1,1}^\alpha \mathbf{u}_1 + \mathbf{V}_{1,3}^\alpha \mathbf{u}_3 + \mathbf{V}_{1,4}^\alpha \mathbf{u}_4 = \mathbf{V}_{1,1}^\alpha \mathbf{u}_1 + \mathbf{V}_{1,3}^\alpha \mathbf{u}_3 + \mathbf{V}_{1,4}^\alpha \mathbf{U} \mathbf{u}_e$$

$$(4.8) \quad \mathbf{v}_2 = \mathbf{V}_{2,2}^\beta \mathbf{u}_2 + \mathbf{V}_{2,3}^\alpha \mathbf{u}_3 + \mathbf{V}_{2,4}^\alpha \mathbf{u}_4 = \mathbf{V}_{2,2}^\beta \mathbf{u}_2 + \mathbf{V}_{2,3}^\alpha \mathbf{u}_3 + \mathbf{V}_{2,4}^\alpha \mathbf{U} \mathbf{u}_e.$$

Equations (4.2) and (4.3) provide two different formulas for  $\mathbf{v}_3$ , either of which could be used. For numerical stability, we use the average of the two:

$$(4.9) \quad \mathbf{v}_3 = \frac{1}{2} (\mathbf{V}_{3,1}^\alpha \mathbf{u}_1 + \mathbf{V}_{3,3}^\alpha \mathbf{u}_3 + \mathbf{V}_{3,4}^\alpha \mathbf{u}_4 + \mathbf{V}_{3,2}^\beta \mathbf{u}_2 + \mathbf{V}_{3,3}^\beta \mathbf{u}_3 + \mathbf{V}_{3,4}^\beta \mathbf{u}_4)$$

$$(4.10) \quad = \frac{1}{2} (\mathbf{V}_{3,1}^\alpha \mathbf{u}_1 + \mathbf{V}_{3,3}^\alpha \mathbf{u}_3 + \mathbf{V}_{3,4}^\alpha \mathbf{U} \mathbf{u}_e + \mathbf{V}_{3,2}^\beta \mathbf{u}_2 + \mathbf{V}_{3,3}^\beta \mathbf{u}_3 + \mathbf{V}_{3,4}^\beta \mathbf{U} \mathbf{u}_e).$$

Combining (4.7) – (4.10) we obtain

$$\begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \end{bmatrix} = \left( \begin{bmatrix} \mathbf{V}_{1,1}^\alpha & \mathbf{0} & \mathbf{V}_{1,3}^\alpha \\ \mathbf{0} & \mathbf{V}_{2,2}^\beta & \mathbf{V}_{2,3}^\beta \\ \frac{1}{2} \mathbf{V}_{3,1}^\alpha & \frac{1}{2} \mathbf{V}_{3,2}^\beta & \frac{1}{2} \mathbf{V}_{3,3}^\alpha + \frac{1}{2} \mathbf{V}_{3,3}^\beta \end{bmatrix} + \begin{bmatrix} \mathbf{V}_{1,4}^\alpha \\ \mathbf{V}_{2,4}^\beta \\ \frac{1}{2} \mathbf{V}_{3,4}^\alpha + \frac{1}{2} \mathbf{V}_{3,4}^\beta \end{bmatrix} \mathbf{U} \right) \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \end{bmatrix}.$$

In other words,

$$(4.11) \quad \mathbf{V} = \begin{bmatrix} \mathbf{V}_{1,1}^\alpha & \mathbf{0} & \mathbf{V}_{1,3}^\alpha \\ \mathbf{0} & \mathbf{V}_{2,2}^\beta & \mathbf{V}_{2,3}^\beta \\ \frac{1}{2} \mathbf{V}_{3,1}^\alpha & \frac{1}{2} \mathbf{V}_{3,2}^\beta & \frac{1}{2} \mathbf{V}_{3,3}^\alpha + \frac{1}{2} \mathbf{V}_{3,3}^\beta \end{bmatrix} + \begin{bmatrix} \mathbf{V}_{1,4}^\alpha \\ \mathbf{V}_{2,4}^\beta \\ \frac{1}{2} \mathbf{V}_{3,4}^\alpha + \frac{1}{2} \mathbf{V}_{3,4}^\beta \end{bmatrix} \mathbf{U}.$$

An analogous computation for  $\mathbf{w}_e$  yields

$$(4.12) \quad \mathbf{W} = \begin{bmatrix} \mathbf{W}_{1,1}^\alpha & \mathbf{0} & \mathbf{W}_{1,3}^\alpha \\ \mathbf{0} & \mathbf{W}_{2,2}^\beta & \mathbf{W}_{2,3}^\beta \\ \frac{1}{2} \mathbf{W}_{3,1}^\alpha & \frac{1}{2} \mathbf{W}_{3,2}^\beta & \frac{1}{2} \mathbf{W}_{3,3}^\alpha + \frac{1}{2} \mathbf{W}_{3,3}^\beta \end{bmatrix} + \begin{bmatrix} \mathbf{W}_{1,4}^\alpha \\ \mathbf{W}_{2,4}^\beta \\ \frac{1}{2} \mathbf{W}_{3,4}^\alpha + \frac{1}{2} \mathbf{W}_{3,4}^\beta \end{bmatrix} \mathbf{U}.$$

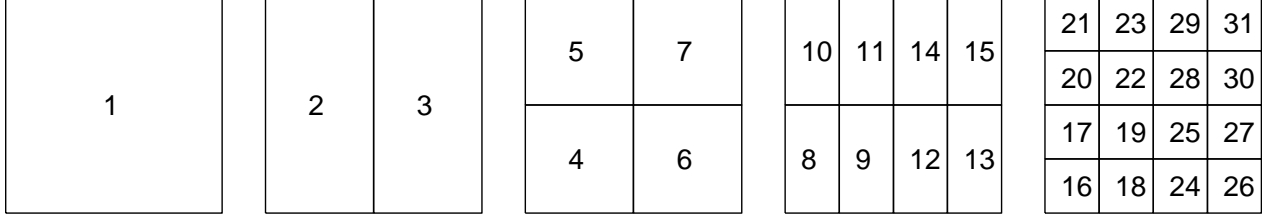


FIGURE 5. The square domain  $\Omega$  is split into  $4 \times 4$  leaf boxes. These are then gathered into a binary tree of successively larger boxes as described in Section 5.1. One possible enumeration of the boxes in the tree is shown, but note that the only restriction is that if box  $\tau$  is the parent of box  $\sigma$ , then  $\tau < \sigma$ .

## 5. THE FULL HIERARCHICAL SCHEME

**5.1. The algorithm.** Now that we know how to construct the DtN operator for a leaf (Section 3), and how to merge the DtN operators of two neighboring patches to form the DtN operator of their union (Section 4), we are ready to describe the full hierarchical scheme for solving (1.1).

First we partition the domain  $\Omega$  into a collection of square (or possibly rectangular) boxes, called *leaf boxes*. These should be small enough that a small spectral mesh with  $p \times p$  nodes (for, say,  $p = 20$ ) accurately interpolates both any potential solution  $u$  of (1.1) and its partial derivatives  $\partial_1 u$ ,  $\partial_2 u$ , and  $-\Delta u$ . Let  $\{\mathbf{x}_k\}_{k=1}^N$  denote the points in this mesh. (Observe that nodes on internal boundaries are shared between two or four local meshes.) Next construct a binary tree on the collection of boxes by hierarchically merging them, making sure that all boxes on the same level are roughly of the same size, cf. Figure 5. The boxes should be ordered so that if  $\tau$  is a parent of a box  $\sigma$ , then  $\tau < \sigma$ . We also assume that the root of the tree (i.e. the full box  $\Omega$ ) has index  $\tau = 1$ .

With each box  $\tau$ , we define two index vectors  $I_e^\tau$  and  $I_i^\tau$  as follows:

$I_e^\tau$  A list of all indices of the nodes on the boundary of  $\tau$ .

$I_i^\tau$  For a leaf  $\tau$ ,  $I_i^\tau$  is a list of all interior nodes of  $\tau$ .

For a parent  $\tau$ ,  $I_i^\tau$  is a list of all its interior nodes that are not interior nodes of its children.

Next we execute a pre-computation in which for every box  $\tau$ , we construct the following matrices:

$\mathbf{U}^\tau$  The matrix that maps the values of  $\mathbf{u}$  on the boundary of a box to the values of  $\mathbf{u}$  on the interior nodes of the box. In other words,  $\mathbf{u}(I_i^\tau) = \mathbf{U}^\tau \mathbf{u}(I_e^\tau)$ .

$\mathbf{V}^\tau$  The matrix that maps the values of  $\mathbf{u}$  on the boundary of a box to the values of  $\mathbf{v}$  (tabulating  $du/dx_1$ ) on the boundary of a box. In other words,  $\mathbf{v}(I_e^\tau) = \mathbf{V}^\tau \mathbf{u}(I_e^\tau)$ .

$\mathbf{W}^\tau$  The matrix that maps the values of  $\mathbf{u}$  on the boundary of a box to the values of  $\mathbf{w}$  (tabulating  $du/dx_2$ ) on the boundary of a box. In other words,  $\mathbf{w}(I_e^\tau) = \mathbf{W}^\tau \mathbf{u}(I_e^\tau)$ .

To this end, we scan all boxes in the tree, going from smaller to larger. For any leaf box  $\tau$ , a dense matrix  $\mathbf{A}^\tau$  of size  $p^2 \times p^2$  that locally approximates the differential operator in (1.1) is formed. Then the matrices  $\mathbf{U}^\tau$ ,  $\mathbf{V}^\tau$ , and  $\mathbf{W}^\tau$  are constructed via formulas (3.3), (3.4), and (3.5). For a parent box  $\tau$  with children  $\sigma_1$  and  $\sigma_2$ , the matrices  $\mathbf{U}^\tau$ ,  $\mathbf{V}^\tau$ , and  $\mathbf{W}^\tau$  are formed from the DtN operators encoded in the matrices  $\mathbf{V}^{\sigma_1}$ ,  $\mathbf{W}^{\sigma_1}$ ,  $\mathbf{V}^{\sigma_2}$ ,  $\mathbf{W}^{\sigma_2}$  using the formulas (4.5), (4.11), and (4.12). The full algorithm is summarized in Figure 6. An illustrated cartoon of the merge process is provided in Appendix A.

Once all the matrices  $\{\mathbf{U}^\tau\}_\tau$  have been formed, it is a simple matter to construct a vector  $\mathbf{u}$  holding approximations to the solution  $u$  of (1.1). The nodes are scanned starting with the root, and then proceeding down in the tree towards smaller boxes. When a box  $\tau$  is processed, the value of  $\mathbf{u}$  is known for all nodes on its boundary (i.e. those listed in  $I_e^\tau$ ). The matrix  $\mathbf{U}^\tau$  directly maps these



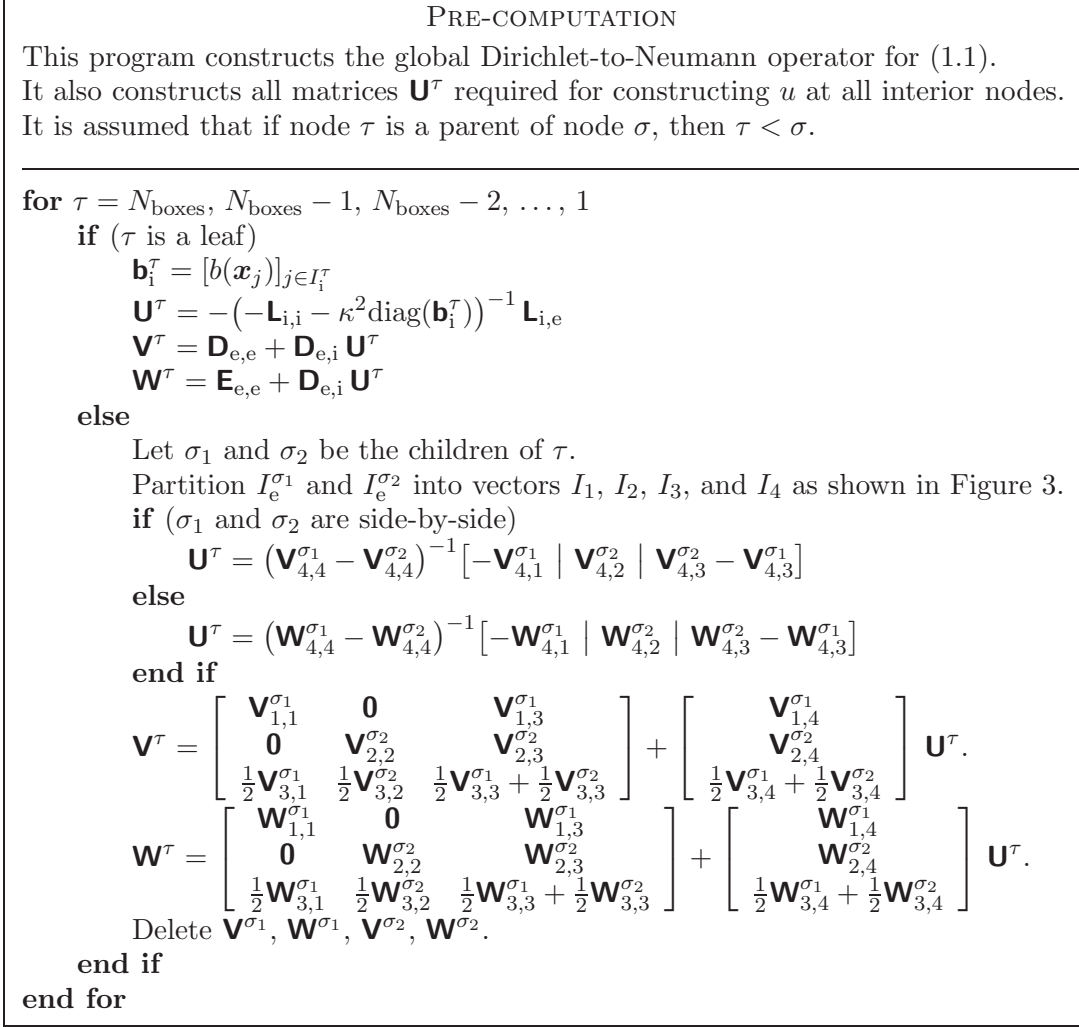


FIGURE 6. Pre-computation

values to the values of  $\mathbf{u}$  on the nodes in the interior of  $\tau$  (i.e. those listed in  $I_i^\tau$ ). When all nodes have been processed, all entries of  $\mathbf{u}$  have been computed. Figure 7 summarizes the solve stage.

**Remark 5.1.** Every interior meshpoint  $\mathbf{x}_k$  belongs to the index vector  $I_i^\tau$  for precisely one node  $\tau$ . In other words  $\bigcup_\tau I_i^\tau$  forms a disjoint union of the interior mesh points.

**Remark 5.2.** The way the algorithms are described, we compute for each node  $\tau$  matrices  $\mathbf{V}^\tau$  and  $\mathbf{W}^\tau$  that allow the computation of both the normal and the tangential derivative at any boundary node, given the Dirichlet data on the boundary. This is done for notational convenience only. In practice, any rows of  $\mathbf{V}^\tau$  and  $\mathbf{W}^\tau$  that correspond to evaluation of tangential derivatives need never be evaluated since tangential derivatives do not enter into consideration at all.

**Remark 5.3.** The merge stage is exact when performed in exact arithmetic. The only approximation involved is the approximation of the solution  $u$  on a leaf by its interpolating polynomial.

**5.2. Complexity analysis.** The analysis of the asymptotic cost of the algorithm in Section 5.1 closely mimics the analysis of the classical nested dissection algorithm [7, 3]. For simplicity, we analyze the simplest situation in which a square domain is divided into  $4^L$  leaf boxes, each holding

<b>SOLVER</b> This program constructs an approximation $\mathbf{u}$ to the solution $u$ of (1.1). It assumes that all matrices $\mathbf{U}^\tau$ have already been constructed in a pre-computation. It is assumed that if node $\tau$ is a parent of node $\sigma$ , then $\tau < \sigma$ .
<hr/> $\mathbf{u}(k) = f(\mathbf{x}_k)$ for all $k \in I_e^1$ . <b>for</b> $\tau = 1, 2, 3, \dots, N_{\text{boxes}}$ $\mathbf{u}(I_i^\tau) = \mathbf{U}^\tau \mathbf{u}(I_i^\tau)$ . <b>end for</b>

FIGURE 7. Solve stage.

a spectral cartesian mesh with  $p \times p$  points. The total number of unknowns in the system is then roughly  $4^L p^2$  (to be precise,  $N = 4^L (p-1)^2 + 2^{L+1} (p-1) + 1$ ).

*Cost of leaf computation:* Evaluating the formulas (3.3), (3.4), and (3.5) requires dense matrix algebra on matrices of size roughly  $p^2 \times p^2$ . Since there are about  $N/p^2$  leaves, the total cost is

$$T_{\text{leaf}} \sim \frac{N}{p^2} \times (p^2)^3 \sim N p^4.$$

*Cost of the merge operations:* For an integer  $\ell \in \{0, 1, 2, \dots, L\}$ , we refer to “level  $\ell$ ” as the collection of boxes whose side length is  $2^{-\ell}$  times the side of the full box  $\Omega$  (so that  $\ell = 0$  corresponds to the root and  $\ell = L$  corresponds to the set of leaf boxes). To form the matrices  $\mathbf{U}^\tau$ ,  $\mathbf{V}^\tau$ , and  $\mathbf{W}^\tau$  for a box on level  $\ell$ , we need to evaluate each of the formulas (4.5), (4.11), and (4.12) three times, with each computation involving matrices of size roughly  $2^{-\ell} N^{0.5} \times 2^{-\ell} N^{0.5}$ . Since there are  $4^\ell$  boxes on level  $\ell$ , we find that the cost of processing level  $\ell$  is

$$T_\ell \sim 4^\ell \times \left(2^{-\ell} N^{0.5}\right)^3 \sim 2^{-\ell} N^{1.5}.$$

Adding the costs at all levels, we get

$$T_{\text{merge}} \sim \sum_{\ell=0}^{L-1} T_\ell \sim \sum_{\ell=0}^{L-1} 2^{-\ell} N^{1.5} \sim N^{1.5}.$$

*Cost of solve stage:* The cost of processing a non-leaf node on level  $\ell$  is simply the cost of a matrix-vector multiply involving the dense matrix  $\mathbf{U}^\tau$  of size  $2^{-\ell} N^{0.5} \times 2^{-\ell} N^{0.5}$ . For a leaf,  $\mathbf{U}^\tau$  is of size roughly  $p^2 \times p$ . Therefore

$$T_{\text{leaf}} \sim \frac{N}{p^2} \times p^2 p + \sum_{\ell=0}^{L-1} 4^\ell \times \left(2^{-\ell} N^{0.5}\right)^2 \sim N p + N L \sim N p + N \log(N).$$

**5.3. Problem of resonances.** The scheme presented in Section 5.1 will fail if one of the patches in the hierarchical partitioning is resonant in the sense that there exist non-trivial solutions to the Helmholtz equation that have zero Dirichlet data at the boundary of the patch. In this case, the Neumann data for the patch is not uniquely determined by the Dirichlet data, and the DtN operator cannot exist. In practice, this problem will of course arise even if we are merely *close* to a resonance, and will be detected by the discovery that the inverse matrix in the formulas (3.3) and (4.5) for the solution operator  $\mathbf{U}^\tau$  is ill-conditioned.

It is our experience from working with domains of size one hundred wave-lengths or less that resonances are very rare; one almost never encounters the problem. Nevertheless, it is important to

monitor the conditioning of the formulas (3.3) and (4.5) to ensure the accuracy of the final answer. Should a problem be detected, the easiest solution would be to simply start the computation over with a different tessellation of the domain  $\Omega$ . Very likely, this will resolve the problem.

Current efforts to formulate variations of the scheme that are inherently not vulnerable to resonances are described in Section 7.5.

## 6. NUMERICAL EXPERIMENTS

This section reports the results of some numerical experiments with the method described in Section 5.1. The method was implemented in Matlab and the experiments executed on a Lenovo W510 laptop with a quad core Intel i7 Q720 processor with 1.6GHz clockspeed, and 16GB of RAM.

The speed and memory requirements of the algorithm were investigated by solving the special case where  $b = 0$  in (1.1), and the Dirichlet data is simply set to equal a known analytic solution. The results are reported in Section 6.1. We also report the errors incurred in the special case, but it should be noted that these represent only a best case estimate of the errors since the equation solved is particularly benign. To get a more realistic estimate of the errors in the method, we also applied it to three situation in which exact solutions are not known: A problem with variable coefficients in Section 6.2, a problem on an L-shaped domain in Section 6.3, and a convection-diffusion problem in Section 6.4.

**6.1. Constant coefficient Helmholtz.** We solved the basic Helmholtz equation

$$(6.1) \quad -\Delta u(\mathbf{x}) - \kappa^2 u(\mathbf{x}) = 0 \quad \mathbf{x} \in \Omega,$$

$$(6.2) \quad u(\mathbf{x}) = f(\mathbf{x}) \quad \mathbf{x} \in \Gamma,$$

where  $\Omega = [0, 1]^2$  and  $\Gamma = \partial\Omega$ . The boundary data were in this first set of experiments chosen as the restriction to  $\Gamma$  of an exact solution

$$(6.3) \quad u_{\text{exact}}(\mathbf{x}) = Y_0(\kappa|\mathbf{x} - \hat{\mathbf{x}}|),$$

where  $\hat{\mathbf{x}} = (-0.2, 0.4)$ , and where  $Y_0$  is the 0'th Bessel function of the second kind. This experiment serves two purposes. The first is to systematically measure the speed and memory requirements of the method described in Section 5.1. The second is to get a sense of what errors can be expected in a “best case” scenario with a very smooth solution. Observe however that the situation is by no means artificial since the smoothness of this case is exactly what one encounters when the solver is applied to a free space scattering problem as described in Section 7.3.

The domain  $\Omega$  was discretized into  $n \times n$  patches, and on each patch a  $p \times p$  Cartesian mesh of Chebyshev nodes was placed. The total number of degrees of freedom is then

$$N = (n(p-1))^2 + 2n(p-1) + 1.$$

We tested the method for  $p \in \{6, 11, 21, 41\}$ . For each fixed  $p$ , the method was executed for several different mesh sizes  $n$ . The wave-number  $\kappa$  was chosen to keep a constant of 12 points per wavelength, or  $\kappa = 2\pi n(p-1)/12$ .

Since the exact solution is known in this case, we computed the direct error measure

$$E_{\text{pot}} = \max_{k: \mathbf{x}_k \in \Omega} |u_{\text{computed}}(\mathbf{x}_k) - u_{\text{exact}}(\mathbf{x}_k)|,$$

where  $\{\mathbf{x}_k\}_{k=1}^N$  is the set of all mesh points. We also computed the maximum error in the gradient of  $u$  on the boundary as computed via the  $\mathbf{V}$  and  $\mathbf{W}$  operators on the root box,

$$E_{\text{grad}} = \max_{k: \mathbf{x}_k \in \Gamma} \{|v_{\text{computed}}(\mathbf{x}_k) - [\partial_1 u_{\text{exact}}](\mathbf{x}_k)|, |w_{\text{computed}}(\mathbf{x}_k) - [\partial_2 u_{\text{exact}}](\mathbf{x}_k)|\}.$$

$p$	$N$	$N_{\text{wave}}$	$t_{\text{inv}}$ (sec)	$t_{\text{solve}}$ (sec)	$E_{\text{pot}}$	$E_{\text{grad}}$	$M$ (MB)	$M/N$ (reals/DOF)
6	6561	6.7	0.28	0.0047	8.02105e-03	3.06565e-01	2.8	56.5
6	25921	13.3	0.96	0.0184	1.67443e-02	1.33562e+00	12.7	64.2
6	103041	26.7	4.42	0.0677	3.60825e-02	5.46387e+00	56.2	71.5
6	410881	53.3	20.23	0.2397	3.39011e-02	1.05000e+01	246.9	78.8
6	1640961	106.7	88.73	0.9267	7.48385e-01	4.92943e+02	1075.0	85.9
11	6561	6.7	0.16	0.0019	2.67089e-05	1.08301e-03	2.9	58.0
11	25921	13.3	0.68	0.0073	5.30924e-05	4.34070e-03	13.0	65.7
11	103041	26.7	3.07	0.0293	1.01934e-04	1.60067e-02	57.4	73.0
11	410881	53.3	14.68	0.1107	1.07747e-04	3.49637e-02	251.6	80.2
11	1640961	106.7	68.02	0.3714	2.17614e-04	1.37638e-01	1093.7	87.4
21	6561	6.7	0.23	0.0011	2.56528e-10	1.01490e-08	4.4	87.1
21	25921	13.3	0.92	0.0044	5.24706e-10	4.44184e-08	18.8	95.2
21	103041	26.7	4.68	0.0173	9.49460e-10	1.56699e-07	80.8	102.7
21	410881	53.3	22.29	0.0727	1.21769e-09	3.99051e-07	344.9	110.0
21	1640961	106.7	99.20	0.2965	1.90502e-09	1.24859e-06	1467.2	117.2
21	6558721	213.3	551.32	20.9551	2.84554e-09	3.74616e-06	6218.7	124.3
41	6561	6.7	1.50	0.0025	9.88931e-14	3.46762e-12	7.9	157.5
41	25921	13.3	4.81	0.0041	1.58873e-13	1.12883e-11	32.9	166.4
41	103041	26.7	18.34	0.0162	3.95531e-13	5.51141e-11	137.1	174.4
41	410881	53.3	75.78	0.0672	3.89079e-13	1.03546e-10	570.2	181.9
41	1640961	106.7	332.12	0.2796	1.27317e-12	7.08201e-10	2368.3	189.2

TABLE 1. Results from an experiment with a constant coefficient Helmholtz problem on a square. The boundary data were picked so that the analytic solution was known; as a consequence, the solution is smooth, and can be smoothly extended across the boundary. The wave-number was chosen to keep a constant of 12 discretization points per wave-length.

Table 1 reports the following variables:

$N_{\text{wave}}$  The number of wave-lengths along one side of  $\Omega$ .

$t_{\text{inv}}$  The time in seconds required to execute the pre-computation in Figure 6.

$t_{\text{solve}}$  The time in seconds required to execute the solve in Figure 7.

$M$  The amount of RAM used in the pre-computation in MB.

The table also reports the memory requirements in terms of number of the number of double precision reals that need to be stored per degree of freedom in the discretization.

The high-order version of the method ( $p = 41$ ) was also capable of performing a high accuracy solve with only six points per wave-length. The results are reported in Table 2.

We see that increasing the spectral order is very beneficial for improving accuracy. However, the speed deteriorates and the memory requirements increase as  $p$  grows. Choosing  $p = 21$  appears to be a good compromise.

**Remark 6.1.** In the course of executing the numerical examples, the instability problem described in Section 5.3 was detected precisely once (for  $p = 16$  and 12 points per wave length).

**6.2. Variable coefficient Helmholtz.** We solved the equation

$$(6.4) \quad -\Delta u(\mathbf{x}) - \kappa^2 (1 - b(\mathbf{x})) u(\mathbf{x}) = 0 \quad \mathbf{x} \in \Omega,$$

$$(6.5) \quad u(\mathbf{x}) = f(\mathbf{x}) \quad \mathbf{x} \in \Gamma,$$

where  $\Omega = [0, 1]^2$ , where  $\Gamma = \partial\Omega$ , and where

$$b(\mathbf{x}) = (\sin(4\pi x_1) \sin(4\pi x_2))^2.$$

$p$	$N$	$N_{\text{wave}}$	$t_{\text{inv}}$	$t_{\text{solve}}$	$E_{\text{pot}}$	$E_{\text{grad}}$	$M$	$M/N$
			(sec)	(sec)			(MB)	(reals/DOF)
41	6561	13.3	1.30	0.0027	1.54407e-09	1.78814e-07	7.9	157.5
41	25921	26.7	4.40	0.0043	1.42312e-08	2.35695e-06	32.9	166.4
41	103041	53.3	17.54	0.0199	1.73682e-08	5.84193e-06	137.1	174.4
41	410881	106.7	72.90	0.0717	2.28475e-08	1.51575e-05	570.2	181.9
41	1640961	213.3	307.37	0.3033	4.12809e-08	5.51276e-05	2368.3	189.2

TABLE 2. This table illustrates the same situation as Table 1, but now  $\kappa$  is increased twice as fast (so that we keep only 6 points per wave-length).

The Helmholtz parameter was kept fixed at  $\kappa = 80$ , corresponding to a domain size of  $12.7 \times 12.7$  wave lengths. The boundary data was given by

$$f(\mathbf{x}) = \cos(8x_1)(1 - 2x_2).$$

The equation (6.4) was discretized and solved as described in Section 6.1. The speed and memory requirements for this computation are exactly the same as for the example in Section 6.1 (they do not depend on what equation is being solved), so we now focus on the accuracy of the method. We do not know of an exact solution, and therefore report pointwise convergence. Letting  $u_N$  denote the value of  $u$  computed using  $N$  degrees of freedom, we used

$$E_N^{\text{int}} = u_N(\hat{\mathbf{x}}) - u_{4N}(\hat{\mathbf{x}})$$

as an estimate for the pointwise error in  $u$  at the point  $\hat{\mathbf{x}} = (0.75, 0.25)$ . We analogously estimated convergence of the normal derivative at the point  $\hat{\mathbf{y}} = (0.75, 0.00)$  by measuring

$$E_N^{\text{bnd}} = w_N(\hat{\mathbf{y}}) - w_{4N}(\hat{\mathbf{y}}).$$

The results are reported in Table 3. Table 4 reports the results from an analogous experiment, but now for a domain of size  $102 \times 102$  wave-lengths.

We observe that accuracy is almost as good as for the constant coefficient case. Ten digits of accuracy is easily attained, but getting more than that seems challenging; increasing  $N$  further leads to no improvement in accuracy. The method appears to be stable in the sense that nothing bad happens when  $N$  is either too large or too small.

**6.3. L-shaped domain.** We solved the equation

$$(6.6) \quad -\Delta u(\mathbf{x}) - \kappa^2 u(\mathbf{x}) = 0 \quad \mathbf{x} \in \Omega,$$

$$(6.7) \quad u(\mathbf{x}) = f(\mathbf{x}) \quad \mathbf{x} \in \Gamma,$$

where  $\Omega$  is the L-shaped domain

$$\Omega = [0, 2]^2 \setminus [1, 2]^2,$$

and where the Helmholtz parameter  $\kappa$  is held fixed at  $\kappa = 40$ , making the domain  $12.7 \times 12.7$  wave-lengths large. The pointwise errors were estimated at the points

$$\hat{\mathbf{x}} = (0.75, 0.75), \quad \text{and} \quad \hat{\mathbf{y}} = (1.25, 1.00),$$

via

$$E_N^{\text{int}} = u_N(\hat{\mathbf{x}}) - u_{4N}(\hat{\mathbf{x}}), \quad \text{and} \quad E_N^{\text{bnd}} = w_N(\hat{\mathbf{x}}) - w_{4N}(\hat{\mathbf{x}}).$$

The results are given in Table 5.

We observe that the errors are in this case significantly larger than they were for square domains. This is presumably due to the fact that the solution  $u$  is singular near the re-entrant corner in the L-shaped domain. (When boundary conditions corresponding to an exact solution like (6.3) are imposed, the method is just as accurate as it is for the square domain.) Nevertheless, the method easily attains solutions with between four and five correct digits.

$p$	$N$	pts per wave	$u_N(\hat{\mathbf{x}})$	$E_N^{\text{int}}$	$w_N(\hat{\mathbf{y}})$	$E_N^{\text{bnd}}$
6	6561	6.28	-2.505791196753718	-2.457e-01	-661.0588680825	-8.588e+03
6	25921	12.57	-2.260084219562163	1.676e-01	7926.8096554095	8.141e+04
6	103041	25.13	-2.427668162910011	1.779e-02	-73484.9989261573	-3.894e+04
6	410881	50.27	-2.445455646843485	1.233e-03	-34547.6403539568	-1.235e+03
6	1640961	100.53	-2.446688310709834	7.891e-05	-33313.0000081604	-7.627e+01
6	6558721	201.06	-2.446767218259172		-33236.7252190062	
11	6561	6.28	-2.500353149793093	-5.375e-02	-27023.0713474340	6.524e+03
11	25921	12.57	-2.446599788642489	1.728e-04	-33547.3621639994	-3.153e+02
11	103041	25.13	-2.446772604281610	-9.465e-08	-33232.0940315585	-4.754e-01
11	410881	50.27	-2.446772509631734	3.631e-10	-33231.6186528531	-7.331e-04
11	1640961	100.53	-2.446772509994819		-33231.6179197169	
21	6561	6.28	-2.448236804078803	-1.464e-03	-32991.4583727724	2.402e+02
21	25921	12.57	-2.446772430608166	7.976e-08	-33231.6118304666	5.984e-03
21	103041	25.13	-2.446772510369452	5.893e-11	-33231.6178142514	-5.463e-06
21	410881	50.27	-2.446772510428384	2.957e-10	-33231.6178087887	-2.792e-05
21	1640961	100.53	-2.446772510724068		-33231.6177808723	
41	6561	6.28	-2.446803898373796	-3.139e-05	-33233.0037457220	-1.386e+00
41	25921	12.57	-2.446772510320572	1.234e-10	-33231.6179029824	-8.940e-05
41	103041	25.13	-2.446772510443995	2.888e-11	-33231.6178135860	-1.273e-05
41	410881	50.27	-2.446772510472872	7.731e-11	-33231.6178008533	-4.668e-05
41	1640961	100.53	-2.446772510550181		-33231.6177541722	

TABLE 3. Results from a variable coefficient Helmholtz problem on a domain of size  $12.7 \times 12.7$  wave-lengths.

$p$	$N$	pts per wave	$u_N(\hat{\mathbf{x}})$	$E_N^{\text{int}}$	$w_N(\hat{\mathbf{y}})$	$E_N^{\text{bnd}}$
21	6561	0.79	0.007680026148649	4.085e-03	3828.84075823538	6.659e+03
21	25921	1.57	0.003595286353011	1.615e+00	-2829.88055527014	-1.791e+02
21	103041	3.14	-1.611350573683137	1.452e+00	-2650.80640712917	-5.951e+03
21	410881	6.28	-3.063762877533994	4.557e-03	3299.72573600854	-7.772e+00
21	1640961	12.57	-3.068320356836451	-7.074e-08	3307.49786015114	9.592e-05
21	6558721	25.13	-3.068320286093162		3307.49776422768	
41	6561	0.79	-0.000213617359480	-1.608e-01	-833.919575889393	-1.006e+03
41	25921	1.57	0.160581838547352	-7.415e-01	171.937456004515	-1.797e+03
41	103041	3.14	0.902057033817060	3.970e+00	1969.187023322940	-1.338e+03
41	410881	6.28	-3.068320045777766	2.405e-07	3307.497852217234	8.792e-05
41	1640961	12.57	-3.068320286282830	(-1.897e-10)	3307.497764294187	(6.651e-8)

TABLE 4. Results from a variable coefficient Helmholtz problem on a domain of size  $102 \times 102$  wave-lengths.

6.4. **Convection diffusion.** We solved the equation

$$(6.8) \quad -\Delta u(\mathbf{x}) - 1000 [\partial_2 u](\mathbf{x}) = 0 \quad \mathbf{x} \in \Omega,$$

$$(6.9) \quad u(\mathbf{x}) = f(\mathbf{x}) \quad \mathbf{x} \in \Gamma,$$

where  $\Omega = [0, 1]^2$ , where  $\Gamma = \partial\Omega$ , and where the boundary data was given by

$$f(\mathbf{x}) = \cos(x_1) e^{x_2}.$$

The equation (6.4) was discretized and solved as described in Section 6.1. Note that this case involves a non-oscillatory solution and does not fit the template (1.1). It is included to illustrate how the spectral method handles a sharp gradient in the solution.

$p$	$N$	pts per wave	$u_N(\hat{\mathbf{x}})$	$E_N^{\text{int}}$	$w_N(\hat{\mathbf{y}})$	$E_N^{\text{bnd}}$
6	19602	12.57	8.969213152495405	2.258e+00	226.603823940515	8.748e+01
6	77602	25.13	6.711091204119065	1.317e-01	139.118986915759	4.949e+00
6	308802	50.27	6.579341284597024	8.652e-03	134.169908546083	3.261e-01
6	1232002	100.53	6.570688999911585		133.843774958376	
11	19602	12.57	6.571117172871830	9.613e-04	133.865552472382	3.851e-02
11	77602	25.13	6.570155895761215	5.154e-05	133.827043929015	5.207e-03
11	308802	50.27	6.570104356719250	1.987e-05	133.821836691967	2.052e-03
11	1232002	100.53	6.570084491282650		133.819785089497	
21	19602	12.57	6.570152809642857	4.905e-05	133.898328735897	7.663e-02
21	77602	25.13	6.570103763348836	1.951e-05	133.821703687416	1.943e-03
21	308802	50.27	6.570084254517955	7.743e-06	133.819760759394	7.996e-04
21	1232002	100.53	6.570076511737839		133.818961147570	

TABLE 5. Results from a constant coefficient Helmholtz problem on an L-shaped domain of size  $12.7 \times 12.7$  wave-lengths.

$p$	$N$	$u_N(\hat{\mathbf{x}})$	$E_N^{\text{int}}$	$w_N(\hat{\mathbf{y}})$	$E_N^{\text{bnd}}$
11	25921	1.987126286905920	-3.191e-04	1255.25512379751	-7.191e-03
11	103041	1.987445414657945	3.979e-13	1255.26231503666	-6.529e-04
11	410881	1.987445414657547	2.455e-12	1255.26296795281	-1.889e-05
11	1640961	1.987445414655092		1255.26298684450	
21	25921	1.987076984861468	-3.684e-04	1255.26075989546	-2.186e-03
21	103041	1.987445414658047	-3.009e-13	1255.26294637880	-4.054e-05
21	410881	1.987445414658348	-2.600e-13	1255.26298691798	-7.881e-08
21	1640961	1.987445414658608		1255.26298699680	
41	25921	1.988004762686629	5.593e-04	1255.26290210213	-8.478e-05
41	103041	1.987445414657579	-9.706e-13	1255.26298687891	-1.178e-07
41	410881	1.987445414658550	-1.237e-12	1255.26298699669	-1.636e-09
41	1640961	1.987445414659787		1255.26298699832	

TABLE 6. Errors for the convection diffusion problem (6.8).

The pointwise errors were estimated at the points

$$\hat{\mathbf{x}} = (0.75, 0.25), \quad \text{and} \quad \hat{\mathbf{y}} = (0.75, 0.00),$$

via

$$E_N^{\text{int}} = u_N(\hat{\mathbf{x}}) - u_{4N}(\hat{\mathbf{x}}), \quad \text{and} \quad E_N^{\text{bnd}} = w_N(\hat{\mathbf{x}}) - w_{4N}(\hat{\mathbf{x}}).$$

The results are given in Table 6. Table 7 reports results from an analogous experiment, but now with the strength of the convection term further increased by a factor of 10.

We observe that the method has no difficulties resolving steep gradients, and that moderate order methods ( $p = 11$ ) perform very well here.

## 7. EXTENSIONS

**7.1. Linear complexity algorithms.** In discussing the asymptotic complexity of the scheme in Section 5.1 it is important to distinguish between the case where  $N$  is increased for a fixed wave-number  $\kappa$ , and the case where  $\kappa \sim N^{0.5}$  to keep the number of degrees of freedom per wavelength constant.

Let us first discuss the case where the wave-number  $\kappa$  is kept fixed as  $N$  is increased. In this situation, the matrices  $\mathbf{U}^T$  will for the parent nodes become highly rank deficient (to finite precision).



$p$	$N$	$u_N(\hat{\mathbf{x}})$	$E_N^{\text{int}}$	$w_N(\hat{\mathbf{y}})$	$E_N^{\text{bnd}}$
21	25921	1.476688750775769	-4.700e-01	13002.9937044202	4.325e+02
21	103041	1.946729131937971	-4.206e-02	12570.4750256324	-7.862e-03
21	410881	1.988785675941193	-1.716e-06	12570.4828877374	-4.900e-03
21	1640961	1.988787391699051	(6.719e-13)	12570.4877875310	(-4.411e-04)
41	25921	2.587008191566030	6.407e-01	13002.1084152522	4.316e+02
41	103041	1.946284950165041	-4.250e-02	12570.4835546978	-2.618e-03
41	410881	1.988785277235741	-2.114e-06	12570.4861729647	-2.127e-03
41	1640961	1.988787391699218		12570.4882994934	

TABLE 7. Errors for a convection diffusion problem similar to (6.8), but now for the even more convection dominated operator  $A = -\Delta - 10\,000\,\partial_2$ .

By factoring these matrices in the pre-computation, the solve phase can be reduced to  $O(N)$  complexity. Moreover, the off-diagonal blocks of the matrices  $\mathbf{V}^\tau$  and  $\mathbf{W}^\tau$  will also be of low numerical rank; technically, they can efficiently be represented in data sparse formats such as the  $\mathcal{H}$ -matrix format [6], or the *Hierarchically Semi-Separable*-format of [5, 13]. This property can be exploited to reduce the complexity of the pre-computation from  $O(N^{1.5})$  to  $O(N)$  in a manner similar to what is done for classical nested dissection for finite-element and finite-difference matrices in [4, 8, 10, 12]. Note that while the acceleration of the solve phase is trivial, it takes some work to exploit the more complicated structure in  $\mathbf{V}^\tau$  and  $\mathbf{W}^\tau$ .

The case where  $\kappa \sim N^{0.5}$  as  $N$  increases is more complicated. In this situation, the numerical ranks of the matrices  $\mathbf{U}^\tau$  and the off-diagonal blocks of  $\mathbf{V}^\tau$  and  $\mathbf{W}^\tau$  will be large enough that no reduction in asymptotic complexity can be expected. However, the matrices are in practice far from full rank, and substantial savings can be achieved by exploiting techniques such as those described in the previous paragraph.

**7.2. General elliptic problems.** The algorithm in Section 5.1 can straight-forwardly be generalized to elliptic equations like

$$\begin{aligned}
& -c_{11}(\mathbf{x})[\partial_1^2 u](\mathbf{x}) - 2c_{12}(\mathbf{x})[\partial_1 \partial_2 u](\mathbf{x}) - c_{22}(\mathbf{x})[\partial_2^2 u](\mathbf{x}) \\
& \quad + c_1(\mathbf{x})[\partial_1 u](\mathbf{x}) + c_2(\mathbf{x})[\partial_2 u](\mathbf{x}) + c(\mathbf{x})u(\mathbf{x}) = 0
\end{aligned}$$

as long as the coefficients are smooth and the leading order operator is elliptic. The only modification required is that in the leaf computation, the matrix representing a spectral approximation of the differential operator be replaced by something like

$$\mathbf{A}^\tau = -\mathbf{C}_{11}\mathbf{D}^2 - 2\mathbf{C}_{12}\mathbf{D}\mathbf{E} - \mathbf{C}_{22}\mathbf{E}^2 + \mathbf{C}_1\mathbf{D} + \mathbf{C}_2\mathbf{E} + \mathbf{C},$$

where  $\mathbf{C}_{11}$  is a diagonal matrix with entries  $c_{11}(\mathbf{x}_j)$  for  $j \in I^\tau$ , etc. Observe that only the leaf computation needs to be modified, the merge steps remain exactly the same. The stability and accuracy of the method of course depends on the signs and relative magnitudes of the coefficient terms, but tentative numerical experiments indicate that the method is effective for a broad range of different problems, including convection-dominated convection-diffusion equations.

**7.3. Free space scattering problems in the plane.** If you know the DtN operator for the inhomogeneous square, you can very rapidly solve an exterior scattering problem as follows. Consider the equation

$$-\Delta u(\mathbf{x}) - \kappa^2(1 - b(\mathbf{x}))u(\mathbf{x}) = f(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^2.$$

Appropriate radiation conditions at infinity are imposed on  $u$ . We assume that  $b$  is compactly supported inside the domain  $\Omega$ , and that  $f$  is supported outside  $\Omega$ . The standard technique is to look for a solution  $u$  of the form

$$u = v + w,$$



where  $v$  is an *incoming field* and  $w$  is the *outgoing field*. The incoming field is defined by

$$(7.1) \quad v(\mathbf{x}) = [\phi_\kappa * f](\mathbf{x}) = \int_{\mathbb{R}^2} \phi_\kappa(\mathbf{x} - \mathbf{y}) f(\mathbf{y}) d\mathbf{y},$$

where  $\phi_\kappa$  is the fundamental solution to the free space Helmholtz problem. Then  $-\Delta v - \kappa^2 v = f$ , and since  $b(\mathbf{x}) = 0$  for  $\mathbf{x} \in \Omega^c$ , we find that the outgoing potential  $w$  must satisfy

$$(7.2) \quad -\Delta w(\mathbf{x}) - \kappa^2 w(\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega^c.$$

Now use the method in Section 5.1 to construct the DtN map  $T$  for the problem

$$-\Delta u(\mathbf{x}) - \kappa^2 u(\mathbf{x}) + \kappa^2 b(\mathbf{x}) u(\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega.$$

Then we know that

$$(7.3) \quad v_n|_\Gamma + w_n|_\Gamma = T(v|_\Gamma + w|_\Gamma),$$

where  $v_n$  and  $w_n$  are normal derivatives of  $v$  and  $w$ , respectively. Now use BIE methods to construct the DtN map  $S$  for the problem (7.2) on the exterior domain  $\Omega^c$ . Then  $w$  must satisfy (7.2).

$$(7.4) \quad w_n|_\Gamma = S w|_\Gamma.$$

Combining (7.3) and (7.4) we find

$$v_n|_\Gamma + S w|_\Gamma = T v|_\Gamma + T w|_\Gamma.$$

In other words,

$$(7.5) \quad (S - T) w|_\Gamma = T v|_\Gamma - v_n|_\Gamma.$$

Observing that both  $v|_\Gamma$  and  $v_n|_\Gamma$  can be obtained from (7.1), and that  $S$  and  $T$  are now available, we see that  $w|_\Gamma$  can be determined by solving (7.5).

**7.4. Problems in three dimensions.** There is no conceptual difficulty in generalizing the method to problems in  $\mathbb{R}^3$ . However, since the fraction of points located on interfaces will increase, the complexity of the pre-computation and the solve stages will be  $O(N^2)$  and  $O(N^{4/3})$ , respectively. The acceleration techniques described in Section 7.1 will likely become very valuable in constructing highly efficient implementations in 3D.

**7.5. Formulating a scheme impervious to resonances.** As discussed in Section 5.3, the fact that the proposed scheme relies on Dirichlet-to-Neumann maps causes problems when the hierarchical partitioning of the domain involves a sub-domain that admits resonant modes. While this issue can be managed quite easily, it would clearly be preferable to formulate a variation of the scheme that is inherently not vulnerable in this regard. One possible approach would be use a so called “total wave” approach suggested by Yu Chen of New York University (private communication). The idea is to maintain for each leaf not an operator that maps Dirichlet data to Neumann data, but rather a collection of matching pairs of Dirichlet and Neumann data (represented as vectors of tabulated values on the boundary). If you know the collection of pairs for two adjacent boxes, you can construct the collection for the union box via a merge procedure similar to the one used in Section 4. This approach appears to not suffer from any problems in the case where one of the involved boxes admits resonant modes, but has the drawback that it would be less amenable to the acceleration technique described in Section 7.1.

## 8. CONCLUSIONS

The paper describes a composite spectral scheme for solving variable coefficient elliptic PDEs with smooth coefficients on simple domains such as squares and rectangles. The method involves a *direct* solver and can in a single sweep solve problems for which state-of-the-art iterative methods require thousands of iterations. High order spectral approximations are used. As a result, potential fields can be computed to a relative precision of about  $10^{-10}$  using twelve points per wave-length or less.

Numerical experiments indicate that the method is very fast. For a problem involving 1.6M degrees of freedom discretizing a domain of size  $100 \times 100$  wavelengths, the pre-computation stage of the direct solver took less than 2 minutes on a laptop. Once the solution operator had been computed, the actual solve that given a vector of Dirichlet data on the boundary constructs the solution at all 1.6M internal tabulation points required only 0.3 seconds. The computed solution had a relative accuracy of  $10^{-9}$ .

The asymptotic complexity of the method presented is  $O(N^{1.5})$  for the construction of the solution operator, and  $O(N \log N)$  for a solve once the solution operator has been created. For a situation where  $N$  is increased while the wave-number is kept fixed, it appears possible to improve the asymptotic complexity to  $O(N)$  for both the pre-computation and the solve stages (see Section 7.1), but such a code has not yet been written.

The method presented has a short-coming in that it is in principle vulnerable to resonances. It relies on a hierarchical partitioning of the domain, and if any one of the boxes in this partitioning is resonant, the method breaks down. In practice, this problem appears to happen very rarely, and can be both detected and remedied if it does occur.

**Acknowledgments:** The author benefitted greatly from conversations with Vladimir Rokhlin of Yale university. Valuable suggestions were also made by Yu Chen (NYU), Adrianna Gillman (Dartmouth), Leslie Greengard (NYU), and Mark Tygert (NYU). The work was supported by the NSF under contracts 0748488 and 0941476, and by the Wenner-gren foundation. Most of the work was conducted during a sabbatical spent at the Department of Mathematics at Chalmers University and at the Courant Institute at NYU. The support from these institutions is gratefully acknowledged.

## REFERENCES

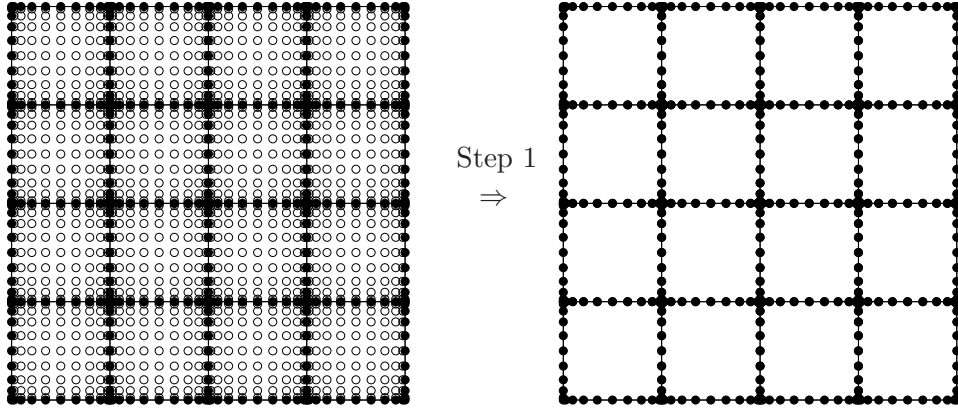
- [1] Yu Chen, *A fast, direct algorithm for the Lippmann-Schwinger integral equation in two dimensions*, Adv. Comput. Math. **16** (2002), no. 2-3, 175–190, Modeling and computation in optics and electromagnetics.
- [2] Ran Duan and Vladimir Rokhlin, *High-order quadratures for the solution of scattering problems in two dimensions*, J. Comput. Physics **228** (2009), no. 6, 2152–2174.
- [3] A. George, *Nested dissection of a regular finite element mesh*, SIAM J. on Numerical Analysis **10** (1973), 345–363.
- [4] A. Gillman, *Fast direct solvers for elliptic partial differential equations*, Ph.D. thesis, Applied Mathematics, University of Colorado at Boulder, 2011.
- [5] Adrianna Gillman, Patrick Young, and Per-Gunnar Martinsson, *A direct solver  $o(n)$  complexity for integral equations on one-dimensional domains*, Frontiers of Mathematics in China **7** (2012), 217–247, 10.1007/s11464-012-0188-3.
- [6] Lars Grasedyck and Wolfgang Hackbusch, *Construction and arithmetics of  $\mathcal{H}$ -matrices*, Computing **70** (2003), no. 4, 295–334.
- [7] A. J. Hoffman, M. S. Martin, and D. J. Rose, *Complexity bounds for regular finite difference and finite element grids*, SIAM J. Numer. Anal. **10** (1973), 364–369.
- [8] Sabine Le Borne, Lars Grasedyck, and Ronald Kriemann, *Domain-decomposition based  $\mathcal{H}$ -LU preconditioners*, Domain decomposition methods in science and engineering XVI, Lect. Notes Comput. Sci. Eng., vol. 55, Springer, Berlin, 2007, pp. 667–674. MR 2334161
- [9] P.G. Martinsson, *A high-order accurate discretization scheme for variable coefficient elliptic pdes in the plane with smooth solutions*, Jan. 2011, arXiv:1101.3383.
- [10] P.G. Schmitz and L. Ying, *A fast direct solver for elliptic problems on general meshes in 2d*, Journal of Computational Physics **231** (2012), no. 4, 1314 – 1338.
- [11] L.N. Trefethen, *Spectral methods in matlab*, SIAM, Philadelphia, 2000.

- [12] Jianlin Xia, Shivkumar Chandrasekaran, Ming Gu, and Xiaoye S. Li, *Superfast multifrontal method for large structured linear systems of equations*, SIAM J. Matrix Anal. Appl. **31** (2009), no. 3, 1382–1411. MR 2587783 (2011c:65072)
- [13] ———, *Fast algorithms for hierarchically semiseparable matrices*, Numerical Linear Algebra with Applications **17** (2010), no. 6, 953–976.

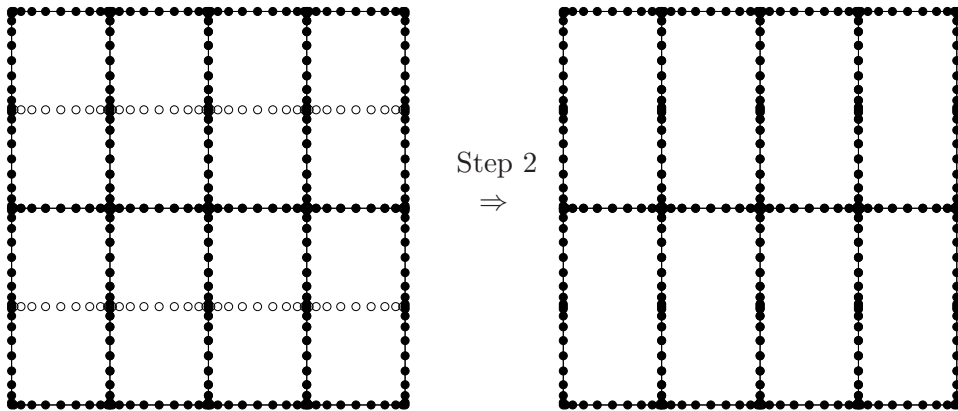
## APPENDIX A. GRAPHICAL ILLUSTRATION OF THE HIERARCHICAL MERGE PROCESS

This section provides an illustrated overview of the hierarchical merge process described in detail in Section 5.1 and in Figure 6. The figures illustrate a situation in which a square domain  $\Omega = [0, 1]^2$  is split into  $4 \times 4$  leaf boxes on the finest level, and an  $8 \times 8$  spectral grid is used in each leaf.

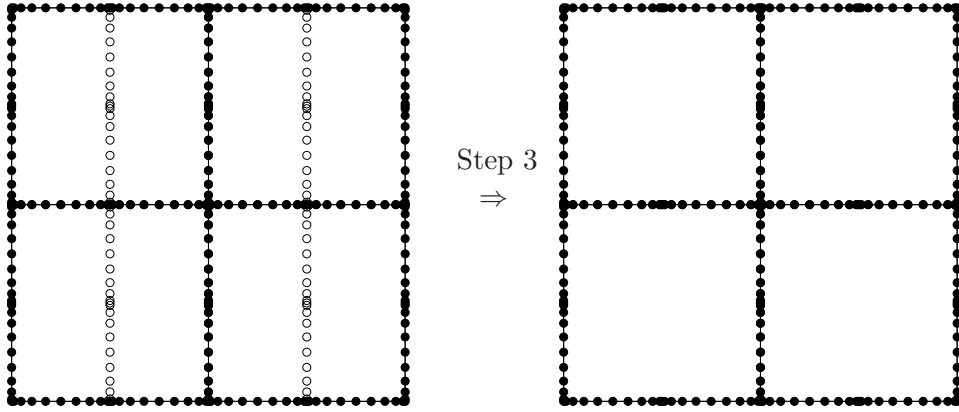
**Step 1:** Partition the box  $\Omega$  into 16 small boxes that each holds an  $8 \times 8$  Cartesian mesh of Chebyshev nodes. For each box, identify the internal nodes (marked in white) and eliminate them as described in Section 3. Construct the solution operator  $\mathbf{U}$ , and the DtN operators encoded in the matrices  $\mathbf{V}$  and  $\mathbf{W}$ .



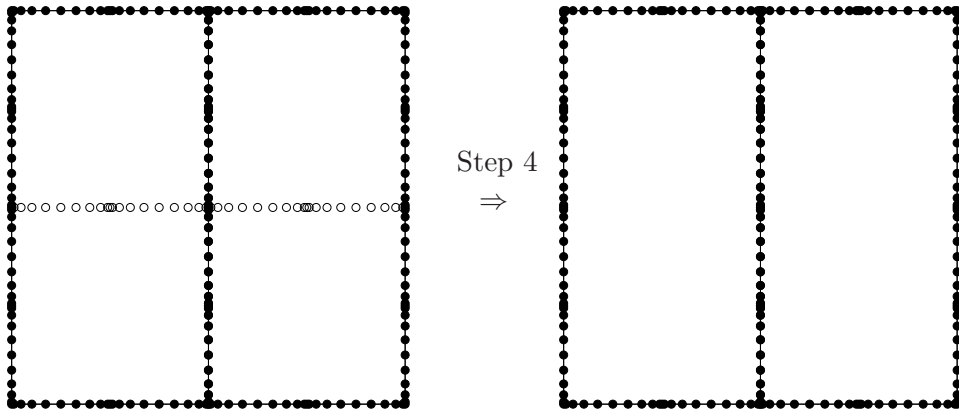
**Step 2:** Merge the small boxes by pairs as described in Section 4. The equilibrium equation for each rectangle is formed using the DtN operators of the two small squares it is made up of. The result is to eliminate the interior nodes (marked in white) of the newly formed larger boxes. Construct the solution operator  $\mathbf{U}$  and the DtN matrices  $\mathbf{V}$  and  $\mathbf{W}$  for the new boxes.



**Step 3:** Merge the boxes created in Step 2 in pairs, again via the process described in Section 4.



**Step 4:** Repeat the merge process once more.



**Step 5:** Repeat the merge process one final time to obtain the DtN operator for the boundary of the whole domain.

